

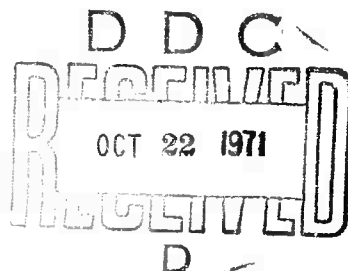
AD 731389



DISTRIBUTION STATEMENT A

Approved for public release;
Distribution Unlimited

TM-4652/500/00



DESCRIPTION OF THE VICENS-REDDY LEXICON

CANDIDATE SELECTION ALGORITHMS

4 October 1971

Reproduced by
NATIONAL TECHNICAL
INFORMATION SERVICE
Springfield, Va. 22151

Sponsored by Advanced Research Projects Agency,
ARPA Order No. 1327

48

TECHNICAL MEMORANDUM

(TM Series)

The work reported was supported by the Advanced Research Projects Agency of the Department of Defense under Contract D-5-67-C-0149, ARPA Order No. 1327, Amendment No. 4, Project No. 2D30, Program Code No. 2P10, "Interactive Systems Research," 15 September 1971 through 15 September 1972.

DESCRIPTION OF THE VICENS-REDDY LEXICON

CANDIDATE SELECTION ALGORITHMS

by

Iris Kameny

H. Barry Ritea

4 October 1971

SYSTEM

DEVELOPMENT

CORPORATION

2500 COLORADO AVE.

SANTA MONICA

CALIFORNIA

M. I. Bernstein, Principal Investigator
Tel. No. (213) 393-9411

90406

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Advanced Research Projects Agency or the U. S. Government.



Distribution of this document is unlimited.

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) System Development Corporation Santa Monica, California		2a. REPORT SECURITY CLASSIFICATION Unclassified	
		2b. GROUP	
3. REPORT TITLE Description of the Vicens-Reddy Lexicon Candidate Selection Algorithms			
4. DESCRIPTIVE NOTES (Type of report and inclusive dates) Technical Report -- July 1970 - September 1971			
5. AUTHOR(S) (First name, middle initial, last name) Iris Kameny H. Barry Ritea			
6. REPORT DATE 4 October 1970		7a. TOTAL NO. OF PAGES 40	7b. NO. OF REFS 0
8a. CONTRACT OR GRANT NO. DAHC15-67-C-0149		9a. ORIGINATOR'S REPORT NUMBER(S) TM-4652/500/00	
b. PROJECT NO. ARPA Order #1327, Amendment #3, Program			
c. Code #1D30, and 1P10		9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)	
d.			
10. DISTRIBUTION STATEMENT Approved for public release; distribution unlimited.			
11. SUPPLEMENTARY NOTES		12. SPONSORING MILITARY ACTIVITY	
13. ABSTRACT This document provides a detailed description of the lexicon candidate-selection algorithms used in the Vicens-Reddy speech recognition system.			

4 October 1971

System Development Corporation
TM-4652/500/00
1
(Page ii. blank)

ABSTRACT

This document provides a detailed description of the lexicon candidate-selection algorithms used in the Vicens-Reddy speech recognition system.

14	KEY WORDS	LINK A		LINK B		LINK C	
		ROLE	WT	ROLE	WT	OLE	WT
	Speech Analysis						
	Candidate Selection						
	Segment Mapping						
	Similarity Evaluation						

TABLE OF CONTENTS

<u>Section</u>	<u>Page</u>
1. INTRODUCTION	1
2. CANDIDATE LIST BUILDING PROCESS	3
2.1 Selection of Initial List of Possible Candidates . . .	3
2.2 Elimination of Candidates with Different Vowel Zero Crossing Characteristics	4
2.3 Elimination of Candidates Having Low Vowel-Similarity Scores with Corresponding Sample Vowel	6
3. CANDIDATE SELECTION PROCESS	6
3.1 Overall Description of the Candidate Selection Process	7
3.2 Segment Mapping Procedure	8
3.2.1 EXPAND Subroutine	9
3.2.2 First Step Vowel and Fricative Mapping	9
3.2.3 Correcting the Vowel Mapping	13
3.2.4 Map Everything	14
3.2.5 Combining in Mapping	14
3.2.6 Update Pointers in R1 and R2 matrices	15
3.2.7 Last Correction Vowel Checking	15
3.2.8 The Similarity Evaluation between R1 and R2	16
3.2.9 Final Evaluation	18
3.3 Final Selection	19
3.4 Error Recovery Procedure	21
3.4.1 Discussion of Changed Heuristics	24
3.5 Similarity Evaluation Procedure	26
3.5.1 Subroutine GRAD	26
3.5.2 Subroutine EVAL	27
REFERENCES	40

4 October 1971

iv

System Development Corporation
TM-4652/500/00

LIST OF TABLES

<u>Table</u>	<u>Page</u>
1. Vowel Dissimilarity Table	5
2. The R1-Matrix	10
3. The R2-Matrix	11
4. The STAMA Table	20

LIST OF FIGURES

<u>Figure</u>	<u>Page</u>
1. The Nine Vowel Categories	5
2. GRAD FORTRAN Program	27

TABLE OF CONTENTS

<u>Section</u>	<u>Page</u>
1. INTRODUCTION	1
2. CANDIDATE LIST BUILDING PROCESS	3
2.1 Selection of Initial List of Possible Candidates . . .	3
2.2 Elimination of Candidates with Different Vowel Zero Crossing Characteristics	4
2.3 Elimination of Candidates Having Low Vowel-Similarity Scores with Corresponding Sample Vowel	6
3. CANDIDATE SELECTION PROCESS	6
3.1 Overall Description of the Candidate Selection Process	7
3.2 Segment Mapping Procedure	8
3.2.1 EXPAND Subroutine	9
3.2.2 First Step Vowel and Fricative Mapping	9
3.2.3 Correcting the Vowel Mapping	13
3.2.4 Map Everything	14
3.2.5 Combining in Mapping	14
3.2.6 Update Pointers in R1 and R2 matrices	15
3.2.7 Last Correction Vowel Checking	15
3.2.8 The Similarity Evaluation between R1 and R2	16
3.2.9 Final Evaluation	18
3.3 Final Selection	19
3.4 Error Recovery Procedure	21
3.4.1 Discussion of Changed Heuristics	24
3.5 Similarity Evaluation Procedure	26
3.5.1 Subroutine GRAD	26
3.5.2 Subroutine EVAL	27
REFERENCES	40

LIST OF TABLES

<u>Table</u>	<u>Page</u>
1. Vowel Dissimilarity Table	5
2. The R1-Matrix	10
3. The R2-Matrix	11
4. The STAMA Table	20

LIST OF FIGURES

<u>Figure</u>	<u>Page</u>
1. The Nine Vowel Categories	5
2. GRAD FORTRAN Program	27

1. INTRODUCTION

This document provides a detailed description of the lexicon candidate-selection algorithms used in the Vicens-Reddy speech recognition system [1]. It is a sequel to SDC TM-4652/200, Description and Analysis of the Vicens-Reddy Pre-processing and Segmentation Algorithms [2], and SDC TM-4652/300, Description and Analysis of the Vicens-Reddy Recognition Algorithms [3], to which the reader is referred for a description of the terms and variables used.

The lexicon candidate-selection process assumes an existing lexicon as described in SDC TM-4652/400, The Lexicon Design for the IBM 360/67 [4], and begins with the feature matrix output of the recognition process. Briefly, the data manipulation steps leading up to the sample feature matrix are:

Step 1: The raw speech data are digitized into 10-msec samples. Each time slice is characterized by six parameters--A1, Z1, A2, Z2, A3, Z3--which represent the amplitude and zero-crossing counts for each of three frequency bands: 150-900 Hz, 900-2200 Hz, and 2200-5000 Hz. All parameters have been smoothed and the amplitude parameters have been normalized with respect to A1 such that the A1 range is 0-63 and the A2, A3 ranges 0-127. One unit of Z1, Z2, or Z3 measurement is equivalent to 50 Hz. The total speech sample represented by n 10-msec segments each containing A1, Z1, A2, Z2, A3, Z3 parameters, plus a closeness computation, is named the Q-matrix.

Step 2: The segmentation routine begins with the Q-matrix and produces a P-matrix of segments containing one or more Q-matrix segments combined on the basis of similarity and closeness.

Step 3: The recognition routine begins with the P-matrix and may, on the basis of certain tests, recombine P-matrix segments. It assigns

linguistic labels to the segments. Recognition then creates the feature matrix or R-matrix from the resulting P-matrix.

The R-matrix:

$r_{1,1}$	$r_{1,2}$	$r_{1,3}$	$r_{1,4}$	$r_{1,5}$	$r_{1,6}$	$r_{1,7}$	$r_{1,8}$	$r_{1,9}$	$r_{1,10}$
TYPR(2)	DURR(2)	AlR(2)	ZlR(2)	A2R(2)	Z2R(2)	A3R(2)	Z3R(2)	SXT(2)	
.
.
.
TYPR(n)	DURR(n)	AlR(n)	ZlR(n)	A2R(n)	Z2R(n)	A3R(n)	Z3R(n)	SXT(n)	

where the first row is reserved for special counts:

- $r_{1,1}$ = number of vowels in the message
- $r_{1,2}$ = number of fricatives in the message
- $r_{1,3}$ = vowel-fricative pattern in binary
- $r_{1,4}$ = last row number
- $r_{1,5}$ = row number of first vowel
- $r_{1,6}$ = row number of second vowel
- $r_{1,7}$ = row number of third vowel
- $r_{1,8}$ = row number of fourth vowel
- $r_{1,9}$ = row number of fifth vowel
- $r_{1,10}$ = unused position of the array

The above array is similar but not identical to the "Construction of the R-matrix" on pp. 25-26 of [3]. The R-matrix described above is the one created and used by CWIPER on the SDC IBM 360/67. The R-matrix described in [3] is the one made by the Vicens-Reddy system on the PDP10.

This document describes a lexicon candidate-selection process which, given a feature matrix of the utterance representation to be recognized, selects a list of possible candidates and from that list chooses the candidate of best match.

2. CANDIDATE-LIST BUILDING PROCESS

Given a feature matrix representation of the speech samples to be recognized, the first task is to select a list of possible candidates from the total lexicon. The subset selection operates serially in the following three stages:

1. Elimination of all candidates whose relative positions of vowels and fricatives are different from those of the sample feature matrix.
2. Elimination of all candidates whose vowel zero-crossing characteristics do not pass similarity tests described below.
3. Elimination of all candidates having low vowel-similarity scores obtained by comparison to those of the sample feature matrix.

2.1 SELECTION OF THE INITIAL LIST OF POSSIBLE CANDIDATES

The vowel-fricative hash* of the sample yields a pointer to a unique entry in the VFHASH table. This entry enables us to begin a series of links to all lexicon entries having the same number of vowels and fricatives as the sample. Before one of these lexicon entries is entered in the initial candidate list, the lexicon entry's vowel-fricative pattern (i.e., the relative positions of vowels and fricatives) is matched against that of the sample. If they match, the lexicon entry is entered in the possible-candidate list.

*For the definition of the vowel-fricative hash, see [4].

If this initial selection of candidates fails (i.e., there are no candidates in the subset), then it is highly likely that the segmentation or recognition algorithms resulted in an erroneous or arbitrary linguistic labeling of one or more segments. An error recovery process is initiated that attempts to change the linguistic labels of segment(s) that might have been incorrectly labeled and to select a list of different candidates. This is discussed in more detail in Section 3.4.

2.2 ELIMINATION OF CANDIDATES WITH DIFFERENT VOWEL ZERO-CROSSING CHARACTERISTICS

In [3] we discussed the assignment of vowel types on the basis of zero-crossing counts in the first and second frequency ranges. To review:

Vowel type = 1 if $Z1 < 6$ and $Z2 < 18$
 = 2 if $Z1 < 6$ and $18 \leq Z2 < 27$
 = 3 if $Z1 < 6$ and $Z2 \geq 27$
 = 4 if $6 \leq Z1 < 9$ and $Z2 < 18$
 = 5 if $6 \leq Z1 < 9$ and $18 \leq Z2 < 27$
 = 6 if $6 \leq Z1 < 9$ and $Z2 \geq 27$
 = 7 if $Z1 \geq 9$ and $Z2 < 18$
 = 8 if $Z1 \geq 9$ and $18 \leq Z2 < 27$
 = 9 if $Z1 \geq 9$ and $Z2 \geq 27$

Diagrammatically, the nine vowel categories appear as in Figure 1.

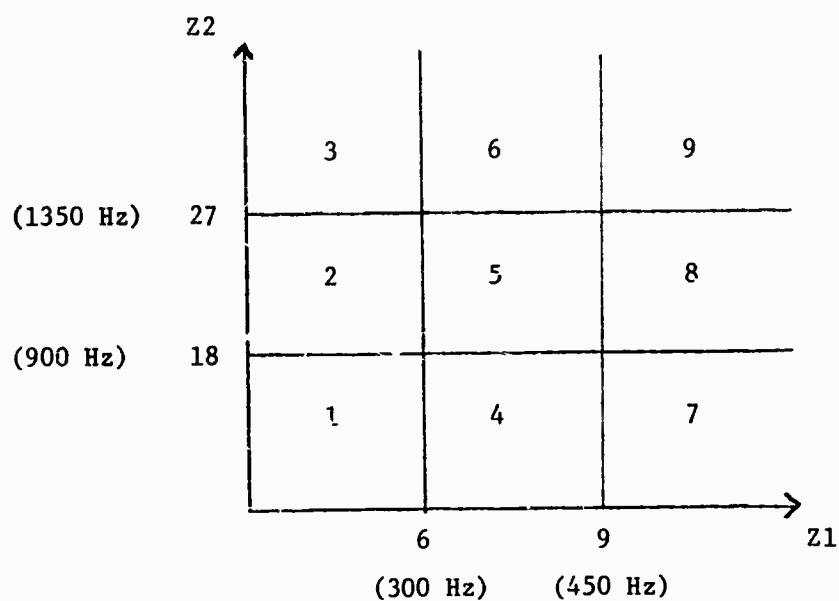


Figure 1. The Nine Vowel Categories

Vicens [1] has devised a table defining rough dissimilarity between pairs of vowels on the basis of their vowel type classification.

Table 1. Vowel Dissimilarity Table
CANDIDATES

Sample	1	2	3	4	5	6	7	8	9
1	1	2	5	3	4	5	0	0	0
2	2	1	2	4	3	4	0	0	0
3	5	2	1	5	4	3	0	0	0
4	3	4	5	1	2	5	3	4	5
5	4	3	4	2	1	2	4	3	4
6	5	4	3	5	2	1	5	4	3
7	0	0	0	3	4	5	1	2	5
8	0	0	0	4	3	4	2	1	2
9	0	0	0	5	4	3	5	2	1

For example a vowel of vowel type 3 and a vowel of vowel type 5 have a crude dissimilarity of 4. Each 0 entry in the table indicates a prohibited correspondence, i.e., if a candidate vowel and a sample vowel are in prohibited correspondence, the candidate is eliminated from the candidate list. The procedure simply looks up in the table the dissimilarity values for all the sample vowels and the vowels of the corresponding candidate. If a prohibited correspondence is detected, the candidate is eliminated. Otherwise, the dissimilarity values are added and form an overall dissimilarity value characterizing the candidate. This process is repeated for all candidates in the candidate list and the list is reordered by increasing order of dissimilarity values. At the end of this process, the lexicon numbers of the candidates are in the STACK table, their corresponding record numbers and beginning word within the record are in the STACK1 table, and the vowel dissimilarity score is in the STACK2 table. Again, if there are no candidates left, the error-recovery routine is initiated.

2.3 ELIMINATION OF CANDIDATES HAVING LOW VOWEL-SIMILARITY SCORES WITH CORRESPONDING SAMPLE VOWELS

For reasons of efficiency, the third attempt at reducing the candidate space is implemented as part of the segment-mapping procedure. It is described in detail in Section 3.2.3. Basically, the procedure computes a similarity score between the vowels of the incoming message and the vowels of each candidate in the list, using the segment-similarity evaluation function described in Section 3.5. If this score is below a threshold, the entry is eliminated from the candidate list; if it is above the threshold, it is retained. Again, if there are no candidates left, the error-recovery routine is initiated.

3. CANDIDATE-SELECTION PROCESS

The previous sections have described how a small list of acceptable candidates can be obtained from a large lexicon by various heuristic procedures. This section discusses how a unique identification of sample feature matrix

can be derived from this list of acceptable candidates by similarity computation. The section is divided into five subsections:

- 3.1 Overall description of the candidate selection process
- 3.2 Segment-mapping procedure
- 3.3 Final selection
- 3.4 Error recovery procedure
- 3.5 Similarity evaluation procedure

3.1 OVERALL DESCRIPTION OF THE CANDIDATE-SELECTION PROCESS

The procedure SERMAP searches the lexicon and computes a "similarity" between the sample feature matrix and all the entries in the acceptable-candidate list. This similarity computation is performed first by calling the segment-mapping procedure, which creates linkages between the segments of the two feature matrices to be matched, and then by averaging the similarity values obtained for each pair of linked segments. The results of this computation are stored for the EVALUT process, which chooses the best-match candidate. If one of the candidates obtains a score greater than or equal to 95 percent, the process immediately stops and returns the candidate print-name (excellent-match-candidate heuristic). As the process continues, modifications of the initial candidate list take place: each time a quite good similarity score ($\geq 80\%$) is obtained, the list is rearranged so as to place next in order all remaining candidates having the same print-name. We assume that if a candidate obtains a score of 80%, it is likely that one of the candidates having the same print-name will obtain 95% or more.

In normal message identification, LISCAN and SERMAP are called each time a new feature matrix of the utterance is built. The initial feature matrix is, of course, the feature matrix determined through the segmentation process. However, since the error-recovery procedure changes the original feature matrix of the utterance, new calls to LISCAN and SERMAP might be necessary to investigate other parts of the lexicon.

The selecting process, which is utilized when no candidate with a very high similarity score is found, is a simple algorithm acting on the scores stored by the matching process. Each candidate left in the candidate list at this point is characterized by three scores:

1. Similarity score for vowels
2. Similarity score for non-vowels
3. Overall similarity score

On the basis of these three numerical values, EVALUT chooses the best-match candidate. The first decision is made on the basis of the overall similarity scores. If the overall scores of several candidates are close, a second decision is made based on the vowel scores. If several candidates are still left after this second stage, the algorithm considers the non-vowel scores.* If more than one candidate is still present in the set of possible responses, the candidate with the best overall score is finally chosen. Of course, the process terminates at any stage when the number of considered candidates reduces to 1. The print-name of the chosen candidate is returned as the recognition response if it satisfies the acceptability criterion.

3.2 SEGMENT-MAPPING PROCEDURE

The mapping procedure is lengthy, complicated and contains many heuristics. We are describing it in detail in order to get a firm grasp on the heuristics involved. This should enable us to make intelligent modifications. Several programming errors have been found and will be noted.

* A program error was detected before the final selection on the basis of the best non-vowel score. The counter K, which was used to count the candidates in the first subset, was not reinitialized before the non-vowel selection, and the input to the final selection could contain candidates previously dropped on the basis of a too-low vowel score.

3.2.1 EXPAND Subroutine

Mapping begins by calling the EXPAND subroutine. EXPAND stores the 10-column R-matrix in a 12-column R1-matrix. This is done so that the original R-matrix can be used to check all remaining potential candidates in the candidate list. The two additional columns are needed to store pointers and weights. The 11th column is labeled BACPT1 and is later renamed POINT1. The 12th column is labeled WT1. EXPAND also takes the current candidate feature matrix from the lexicon and stores this in the so-called R2-matrix. R1 and R2 are described graphically in Tables 2 and 3.

3.2.2 First-Step Vowel and Fricative Mapping

A MAXLNG is set for later usage. This is either the longest segment duration of the R1 and R2 matrices or 24 (whichever is greatest) plus an additional 8. (We are not aware of the reason for the 24 or 8.) All A1 parameters having a zero value in the R1 and R2 matrices are set to 1. The BACPT1, WT1, BACPT2, and WT2 columns are initialized to zero.

We now want to synchronize the comparisons of rows of the R1 matrix with the rows of the R2 matrix. To do this, two columns, MAPAR1 and MAPAR2, are defined and used as follows:

$$\text{MAPAR1}(i) = j \text{ and } \text{MAPAR2}(i) = k$$

means that row j of the R1 matrix is mapped to row k of the R2 matrix.

MAPAR1 and MAPAR2 are initialized as follows:

$$\text{MAPAR1}(1) = 1$$

$$\text{MAPAR1}(2) = 1 + (\text{last row number of R1})$$

$$\text{MAPAR1}(3) = 0$$

$$\vdots$$

$$\text{MAPAR1}(60) = 0$$

$$\text{MAPAR2}(1) = 1$$

Table 2. The R1-Matrix

The R1 Matrix is the feature matrix of the sample:

R1(1,1)	R1(1,2)	R1(1,3)	R1(1,4)	R1(1,5)	R1(1,6)	R1(1,7)	R1(1,8)	R1(1,9)	R1(1,10)	R1(1,11)	R1(1,12)
TYPE1(2)	DUR1(2)	A11(2)	Z11(2)	A21(2)	Z21(2)	A31(2)	Z31(2)	SXT1(2)	IMAGE1(2)	BACPT1(2) or POINT1(2)	WT1(2)
.
.
.
TYPE1(60)	DUR1(60)	A11(60)	Z11(60)	A21(60)	Z21(60)	A31(60)	Z31(60)	SXT1(60)	IMAGE1(60)	BACPT1(60)	WT1(60)

and
R1(1,1) = VOWCT1 the number of vowels in the R1 feature matrix
R1(1,2) = FRIC11 the number of fricatives in the R1 feature matrix
R1(1,3) = PATT1 the vowel-fricative pattern of the R1 feature matrix
R1(1,4) = ROWCT1 the number of rows in the R1 feature matrix including Row 1
R1(1,5) = Row Number of the first vowel
R1(1,6) = Row Number of the second vowel
R1(1,7) = Row Number of the third vowel
R1(1,8) = Row Number of the fourth vowel
R1(1,9) = Row Number of the fifth vowel
R1(1,10) = R1(1,12) are not used

Table 3. The R2-Matrix
The R2 Matrix is the feature matrix of the candidate:

R2(1,1)	R2(1,2)	R2(1,3)	R2(1,4)	R2(1,5)	R2(1,6)	R2(1,7)	R2(1,8)	R2(1,9)	R2(1,10)	R2(1,11)	R2(1,12)
TYPE2(2)	DUR2(2)	A12(2)	Z12(2)	A22(2)	Z22(2)	A32(2)	Z32(2)	SXT2(2)	IMAGE2(2)	BACPT2(2) or POINT2(2)	WT2(2)
.
.
.
TYPE2(60)	DUR2(60)	A12(60)	Z12(60)	A22(60)	Z22(60)	A32(60)	Z32(60)	SXT2(60)	IMAGE2(60)	BACPT2(60)	WT2(60)

and R2(1,1) = VOWCT2 the number of vowels in the R2 feature matrix
R2(1,2) = FRICT2 the number of fricatives in the R2 feature matrix
R2(1,3) = PATT2 the vowel-fricative pattern of the R2 feature matrix
R2(1,4) = ROWCT2 the number of rows in the R2 feature matrix including Row 1
R2(1,5) = Row Number of the first vowel
R2(1,6) = Row Number of the second vowel
R2(1,7) = Row Number of the third vowel
R2(1,8) = Row Number of the fourth vowel
R2(1,9) = Row Number of the fifth vowel
R2(1,10) = R2(1,12) are not used

MAPAR2(2) = 1 + (last row number of R2)
MAPAR2(3) = 0
 : :
 : :
MAPAR2(60) = 0

The R1 matrix and then the R2 matrix are searched for vowels. Each time a vowel is found, its row number is entered in the next available location in the respective MAPAR1 (for R1 vowels) or MAPAR2 (for R2 vowels) table. If the numbers of vowel entries in MAPAR1 and MAPAR2 are different, the message

MAPPING ERROR NUMBER 1 CANDIDATE LEX NO XXX

appears on the user's terminal and this candidate is removed from the list. The same process as described above for vowels is then applied to fricative segments. If the numbers of fricatives in MAPAR1 and MAPAR2 are different, the message

MAPPING ERROR NUMBER 2 CANDIDATE LEX NO XXX

appears on the user's terminal and this candidate is removed from the list.

The sort is then called with the parameter MAPARS. MAPARS is the count of the entries in the MAPAR1 table or the MAPAR2 table (i.e., each table has the same number of entries). The sort does the following:

1. Sorts MAPAR1 entries into ascending order,
i.e., MAPAR1(1) = 1
MAPAR1(2), ..., MAPAR1(MAPARS-1) = row numbers of vowel
and fricative rows in ascending order
MAPAR1(MAPARS) = (LAST ROW NUMBER OF R1) + 1
2. Each time a position change is made for a MAPAR1 entry, makes a corresponding position change for the corresponding entry in MAPAR2. MAPAR2 is ordered by position and not by the intrinsic value of its entries.

3. Sets the BACPT1 column of the R1 matrix: for $i = 1, \dots, \text{MAPARS}$, we set $\text{BACPT1}(\text{MAPAR1}(i)) = i$

Sets the BACPT2 column of the R2 matrix: for $i = 1, \dots, \text{MAPARS}$, we set $\text{BACPT2}(\text{MAPAR2}(i)) = i$

4. Last tests the BACPT2 column for entries in ascending order. If the test fails, the message

MAPPING ERROR NUMBER 3 CANDIDATE LEXNO XXX

appears on the user's terminal and this candidate is removed from the list.

The sort completes the first crude mapping of vowels and fricatives of the sample with the candidate. For any segment j of the R1 matrix, if $\text{BACPT1}(j) \neq 0$, then $\text{MAPAR2}(\text{BACPT1}(j)) =$ segment number of the R2 matrix mapped to segment j . Likewise, for any segment k of the R2 matrix, if $\text{BACPT2}(k) \neq 0$, then $\text{MAPAR1}(\text{BACPT2}(k)) =$ segment number of the R1 matrix mapped to segment k .

3.2.3 Correcting the Vowel Mapping

This routine accomplishes two purposes. First, if a vowel is preceded or followed by a high consonant sound like /r/ or /l/, this consonant may be incorrectly classified VOWEL, thus creating a mislinkage at this early state of the mapping. The same mislinkage may occur if the vowel is a diphthong, in which case either part of it can be classified VOWEL. To correct this defect, this procedure redefines the links on the basis of similarity of parameters between the linked vowel segments and the consonant or nasal segments adjacent to them. The similarity of parameters between segments is defined by the similarity function, which is described in Section 3.5. Second, the mapping procedure continues by computing a similarity score between all the correctly mapped vowels. If the obtained score is below a heuristic threshold (i.e., the number of vowels $\cdot (-6) >$ obtained similarity score), the candidate is eliminated from the candidate

list. This is the third candidate-space-reduction procedure mentioned in Section 2.3.

3.2.4 Map Everything

The program now proceeds by mapping the segments between any two pairs of mapped segments on the basis of parameter similarity. Additional weighting is given to the similarity parameter for special conditions (i.e., when both segments are vowel or both segments are stop or both have the same displacement from the beginning or end of the closest mapped segment). This process is recursively repeated until the program cannot effect any more mapping.

3.2.5 Combining in Mapping

As a result of mapping everything, the few remaining unmapped segments are then candidates for combination with their preceding or following segments. Since these second-order combinations, in general, degrade both representations to be matched, care must be exercised in applying them. The closeness index between segments is computed using the PROXB function, which is the same as the PROXIM function defined in the segmentation procedure [2] except that it uses the R1 or R2 matrix rather than the P-matrix. On the basis of the closeness values between the unmapped segment and its adjacent segments, the closer adjacent segment is chosen. If the closeness value between the unmapped segment and the chosen segment is high enough, a combination occurs. The combinations are done one at a time and in parallel on both representations. Each time a combination occurs the mapping process is re-entered in an attempt to map the segment result of the combining. This mapping-combining process is recursively repeated until no more combining or mapping can be performed.*

*The combining is a complicated process and is accomplished by MAP's calling subroutine COMBN and COMBN's calling the COMPRX and MX subroutines. An error was detected in the COMPRX subroutine in that BACPT1 and BACPT2 were used as if they pointed to the mapped segment in the other matrix directly, instead of indirectly through the MAPAR1 and MAPAR2 tables.

3.2.6 Update Pointers in R1 and R2 Matrices

The pointers in the R1 and R2 matrices are updated as follows:

for $i = 2, \dots, \text{MAPARS}-1$, we set

$\text{POINT1}(\text{MAPAR1}(i)) = \text{MAPAR2}(i)$

$\text{POINT2}(\text{MAPAR2}(i)) = \text{MAPAR1}(i)$

It is important to note that this is not just an updating, as [1] implies, but a redefinition of the pointers. Note that POINT1 is the BACPT1 column of R1 and POINT2 is the BACPT2 column of R2. Previously, BACPT1(j) pointed to an entry in the MAPAR2 table that contained the segment number of the R2 matrix that was mapped to the i^{th} segment in the R1 matrix. Now $\text{POINT1}(j) =$ the segment number of the R2 matrix that is mapped to the j^{th} segment of the R1 matrix. Similarly, $\text{POINT2}(k) =$ the segment number of the R1 matrix that is mapped to the k^{th} segment of the R2 matrix.

3.2.7 Last Correction Vowel Checking

A last attempt to correct the vowel mapping is now made. This correction is based on a comparison between the weighted amplitudes and durations of the mapped vowel segments and the weighted amplitudes and durations of the mapped segments preceding and following the vowel segments if (1) such segments exist and (2) they are transitionals or consonants, or nasals. Given an i segment in R1 and a j segment in R2, the weighting heuristic is as follows:*

$$8 \cdot A11(i) + 4 \cdot A21(i) + 2 \cdot A31(i) + \text{DUR1}(i) + 8 \cdot A12(j) + 4 \cdot A22(j) + 2 \cdot A32(j) + \text{DUR2}(j)$$

A special weight of 60 is added to the weight for a mapped vowel to a mapped vowel.

* This weighting of amplitudes is explained in [2].

3.2.8 The Similarity Evaluation Between R1 and R2

The similarity evaluation procedure computes the following three scores between R1 and R2: the vowel similarity score, the non-vowel similarity score, and the overall similarity score. The vowel similarity score is a sum of the similarity scores of all of the mapped vowels. It is computed by calling the EVAL subroutine, which returns with a similarity score between two mapped segments. A proportion of this score is added to the vowel similarity score. The proportion is determined from the weights of the mapped segments. The weights represent the ratio of the vowel segment duration to the total vowel duration for both R1 and R2. The non-vowel similarity score is computed in a like fashion except that (1) the non-vowel weights represent a proportion of the non-vowel duration to the total non-vowel duration for R1 and R2 and (2) the non-vowel similarity score is corrected by the sum of the weights of the unmapped segments. This correction is necessary only for non-vowels because the occurrence of unmapped vowels would have resulted in an unacceptable candidate before mapping proceeded very far. The overall similarity score is computed from (1) the vowel similarity score weighted by the ratio of the sum of the vowel durations to the total duration for both R1 and R2 and (2) the non-vowel similarity score weighted by the ratio of the sum of the non-vowel durations to the total duration for both R1 and R2. A correction factor representing the proportion of difference in duration (between R1 and R2) to the total duration of R1 and R2 is subtracted from the result. A detailed description of the steps in this evaluation process is given below.

The procedure begins with an examination of the first segments of R1 and R2. If they are both stops, their respective durations are set to $\frac{\text{duration}+1}{2}$.

The last segments of R1 and R2 are handled similarly. R1 and R2 are then examined for unmapped segments. If an unmapped transitional segment is found, its duration is set to zero. If an unmapped burst segment is found, its duration is set equal to $\frac{\text{duration}+1}{2}$.

The vowel duration sum, VSUM, is set equal to the sum of all the vowel durations of R1 and R2. The non-vowel duration sum, RSUM, is set equal to the sum of all the non-vowel durations of R1 and R2. The total duration sum, ASUM, is the sum of all the segment durations of R1 and R2. The vowel weight, VWT is set equal to $\frac{100 \cdot \text{VSUM}}{\text{ASUM}}$. The non-vowel weight, RWT is set equal to $100 - \text{VWT}$.

The weight of each segment in the R1 and R2 matrices is set as follows:

For $i = 2, \dots, \text{ROWCT1}$,

if $\text{TYPE1}(i) = \text{vowel}$, then $\text{WT1}(i) = \frac{\text{DUR1}(i) \cdot 1000}{\text{VSUM}}$

or if $\text{TYPE1}(i) \neq \text{vowel}$, then $\text{WT1}(i) = \frac{\text{DUR1}(i) \cdot 1000}{\text{RSUM}}$;

and for $i = 2, \dots, \text{ROWCT2}$

if $\text{TYPE2}(i) = \text{vowel}$, then $\text{WT2}(i) = \frac{\text{DUR2}(i) \cdot 1000}{\text{VSUM}}$

or if $\text{TYPE2}(i) \neq \text{vowel}$, then $\text{WT2}(i) = \frac{\text{DUR2}(i) \cdot 1000}{\text{RSUM}}$

The vowel similarity score, ANSWE1, and the non-vowel similarity score, ANSWE2, are initialized to zero. The next step is to proceed through the R1 matrix and compute the similarity between each R1 mapped segment and its R2 segment using the subroutine EVAL (see Section 3.5.2).

For $i = 2, \dots, \text{ROWCT1}$

If $\text{POINT1}(i) \neq 0$ (i.e., $\text{POINT1}(i)$ is the R2 segment)

set $J1 = \text{POINT1}(i)$ then call EVAL ($i, J1$)

Set $\text{POINT1}(i) = \text{score}$ (i.e., similarity score between i and $J1$ from EVAL) and set $\text{SCORE} = \text{SCORE} \cdot (\text{WT1}(i) + \text{WT2}(J1))$

Then if $\text{TYPE1}(i) = \text{vowel}$, set $\text{ANSWE1} = \text{ANSWE1} + \text{SCORE}$

or if $\text{TYPE1}(i) \neq \text{vowel}$, set $\text{ANSWE2} = \text{ANSWE2} + \text{SCORE}$

then set $\text{POINT2}(J1) = \frac{\text{SCORE}}{100}$.

The variable, CONSUM, is set equal to the sum of the weights of the unmapped segments in R1 and R2.

Then:

$$\text{CONSUM} = \max(0, \text{CONSUM}-60)$$

$$\text{ANSWE1} = \frac{\text{ANSWE1}}{1000}$$

$$\text{ANSWE2} = - \left(\frac{\text{CONSUM} \cdot 100 - \text{ANSWE2}}{1000} \right)$$

$$\text{ANSWE3} = \sum_{i=2}^{\text{ROWCT1}} \text{DUR1}(i) - \sum_{j=2}^{\text{ROWCT2}} \text{DUR2}(j)$$

then

$$\text{ANSWE3} = \max \left(\frac{|\text{ANSWE3}| \cdot 100}{\text{ASUM}} - 10, 0 \right)$$

3.2.9 Final Evaluation

Set LIM2 = 35 and LIM1 = 40. However, if the sum of vowels and fricatives in the sample is 1, then $\text{LIM1} = \text{LIM1} + \frac{\text{VWT}}{4}$.

Then if:

$\text{ANSWE1} \geq \text{LIM1}$ and $\text{ANSWE2} \geq \text{LIM2}$ and $\text{ANSWE3} \leq 25$ or $\text{ANSWE1} \geq 80$ and either $\text{ANSWE2} \leq \text{LIM2}$ or $\text{ANSWE3} > 25$ and the sum of the vowels and fricatives is 1, then $\text{REPLY} = \frac{(\text{ANSWE1} \cdot \text{VWT}) + (\text{ANSWE2} \cdot \text{RWT})}{100} - \text{ANSWE3}$.

Otherwise the candidate is not acceptable, $\text{REPLY} = 0$, and we return to SERMAP.

If $\text{REPLY} < 50$ and the sum of the fricatives and vowels in the sample is greater than 1, the candidate is not acceptable, $\text{REPLY} = 0$, and we return to SERMAP.

In order to continue the mapping process, either REPLY must be ≥ 50 or REPLY < 50 and the sum of vowels and fricatives must be equal to one.

If there are already 39 acceptable candidates (i.e., INDEX = 39) in the STAMA table, EVALUT is called to select the best candidate. The acceptable candidate count (INDEX) is stepped by one and the current candidate information (i.e., 40 characters of the print name, ANSW1, ANSW2, ANSW3, REPLY) are stored in the INDEX row of the STAMA table (see Table 4). The mapping of this candidate is completed and successful and MAP returns to SEFMAP.

3.3 FINAL SELECTION

Mapping returns to SERMAP with REPLY set to the overall similarity score. If the REPLY > 0 , the lexicon number and record number and beginning word number of the sample are stored in the INDEX row of STAMA. If the REPLY ≥ 95 , the current candidate is the selected candidate and a message is printed on the user's terminal:

```
"YOU SAID (PRINT NAME OF CANDIDATE)"  
"LEXNO__ SESSNO__ SAMPLE__ MANNO__ SCORE__"  
"PREPR__ SEGMNT__ RECOGN__ MAPPI__"  
"CANDIDATES IN LIST WITH SAME NAME__"
```

CWIPER then asks if the sample is to be inserted in the lexicon. If the answer is yes, the sample is inserted. The program then returns to the main driver where a new sample may be analyzed or learned or the program stopped.

If $80 \leq \text{REPLY} < 95$, the remaining candidates in the stack (if there are more than one) are rearranged so as to place next in order all remaining candidates having the same print name as the sample. The mapping routine is again entered to consider the next candidate in the stack.

Table 4. The STAMA Table
THE STAMA TABLE IS THE LIST OF QUALIFYING CANDIDATES
WITH THEIR SIMILARITY SCORES

STAMA(1,1),..., STAMA(1,10)	STAMA(1,11) ANSW1(1)	STAMA(1,12) ANSW2(1)	STAMA(1,13) ANSW3(1)	STAMA(1,14) FACTM1(1)	STAMA(1,15) ANSWE(1)	STAMA(1,16) ADDRESS(1)	STAMA(1,17) AVERAG(1)	STAMA(1,18) Record No. & Beginning Word
First 40 characters of the print name

	ANSW1(39)	ANSW2(39)	ANSW3(39)	FACTM1(39)	ANSWE(39)	ADDRESS(39)	AVERAG(39)	

ANSW1_i = vowel similarityANSW2_i = nonvowel similarity (mapped)ANSW3_i = duration similarityFACTM1_i = count of number of candidates with same print nameANSWE_i = overall similarityADDRESS_i = LXALL addressAVERAG_i : appeared in the original program but is an unused variableSTAMA_{i,18} = record no. and beginning word no.

If the stack is exhausted before a candidate with an overall similarity score ≥ 95 is found, then EVALUT is called to select the best candidate from the STAMA table and enter it in row 1 of the STAMA table. A test (given below) is then made to determine whether this selected candidate is similar enough to be chosen.

Given data arrays:

ENTRY NUMBER	DATA ARRAYS	
	ANS1	ANS4
1	72	80
2	68	80
3	63	80
4	58	80

Let $N1 = \text{MIN}(4, \text{VOWCT1})$. Then if the selected candidate's overall similarity score $\geq \text{ANS4}(N1)$, the vowel similarity score $\geq \text{ANS1}(N1)$, and the non-vowel similarity score $\geq \text{ANS1}(N1)$, the candidate is the selected candidate and the process discussed in Section 3.2.9 occurs. If the test is failed, then we proceed to the error-recovery routine discussed in the following section. However, keep in mind that the current candidate information is in the first row of the STAMA table and (if no other adequate candidate is found) may be the final choice.

3.4 ERROR RECOVERY ROUTINE

The error recovery routine may be entered whenever the candidate list has been exhausted before an acceptable candidate is found. It is based on the assumption that a segmenting or labeling error in the sample will lead to an erroneous set of lexicon candidates. The sample is therefore examined for borderline cases of vowels (a nasal, consonant or transitional might have been classified as a vowel or a weak vowel might have been classified

as a transitional, consonant or nasal) and borderline cases of fricatives (an unvoiced fricative might have been classified as a burst or a voiced fricative might have been classified as a fricative). A feasibility value is assigned to the borderline case and the borderline case list is arranged in decreasing order of feasibility, so that the entries most likely to be incorrectly classified appear first in the list.

Vicens ran the error recovery routine either until 15 seconds had elapsed, or until the first three borderline cases and their combinations had been run, or until a candidate with an overall similarity score ≥ 95 was found. (We discovered that the version of the program we had did not cycle through the three borderline cases and their combinations properly, and we reprogrammed this section of it.)

Assuming that three borderline segments 1,2,3 have been detected in the sample, the following actions are performed:

- Modify 1 and run candidate-list building and selection
- Modify 2 and run candidate-list building and selection
- Modify 3 and run candidate-list building and selection
- Modify 1 and 2 and run candidate-list building and selection
- Modify 1 and 3 and run candidate-list building and selection
- Modify 2 and 3 and run candidate-list building and selection
- Modify 1, 2 and 3 and run candidate-list building and selection

The actual heuristics to determine borderline cases in the program were more complicated than those described in [1] and are discussed in Section 3.4.1. When the selected changes and their combinations have been run, a best candidate may have been found (either as a result of the error recovery or from the initial selection) whose overall similarity score < 95 . Information about this candidate would be in row 1 of the STAMA table. If a candidate had been found with an overall similarity score ≥ 95 , the

error-recovery procedure would have stopped and that candidate would have been selected.

(If no acceptable candidate had been found after running the error-recovery routine, the user would receive the terminal message "No acceptable candidate, even after modifying the sample.") If there is a best candidate whose overall similarity score < 95 , the candidate's scores are now put through a series of acceptance tests before the candidate is actually accepted.

Given the following data arrays:

Entry Number	DATA ARRAY NAME							
	ANS1	ANS2	ANS3	ANS4	ANS5	ANS6	ANS7	ANS8
1	72	65	145	80	60	90	128	60
2	68	60	140	80	58	86	125	60
3	63	55	135	80	56	82	123	60
4	58	50	130	80	54	78	120	60

Let $N1 = \text{MIN}(4, \text{VOWCT})$. $N1$ is the entry number for the data arrays described above. In the following tests all references to scores will be to those belonging to the best candidate and are located in row 1 of the STAMA table.

The tests are performed in the following order:

1. If the overall similarity score $\geq \text{ANS1}(N1)$ and the vowel similarity score $\geq \text{ANS2}(N1)$, the candidate is accepted.
2. If the overall similarity score $\geq \text{ANS2}(N1)$ and the overall similarity score + the vowel similarity score $\geq \text{ANS3}(N1)$, the candidate is accepted.

3. If the overall similarity score $\geq \text{ANS2}(N1)$, and if either the overall similarity score + the vowel similarity score $\geq \text{ANS7}(N1)$ or the non-vowel similarity score $\geq \text{ANS4}(N1)$, and if the vowel similarity score + the non-vowel similarity score + 10 $\geq \text{ANS7}(N1)$, and if the non-vowel similarity score $\geq \text{ANS8}(N1)$, the candidate is accepted.
4. If the overall similarity score $\geq \text{ANS5}(N1)$ and if the overall similarity score $\geq \text{ANS6}(N1)$, the candidate is accepted.

Test 4 above is superfluous, for if one examines the data arrays he can see that in order for the overall similarity score to be $\geq \text{ANS6}(i)$ (which is the ultimate test) it would have to be $\geq \text{ANS2}(i)$ (for $i = 1, 2, 3$ or 4) and therefore would never have reached test 4. Perhaps the data arrays were originally set to different values and changed heuristically without revising the program.

3.4.1 Discussion of Changed Heuristics

Pages 115-116 of [1] discuss the heuristics found useful in defining the borderline cases in error recovery. These differ from the heuristics found in the program in the treatment of the duration parameter. The feasibility value of the transitional, nasal or consonant becoming a vowel is given by:

Reference [1]: $A1 + A2 + A3 + \text{DURATION}/20 - 90$

Computer program: $90 - 5 \cdot \text{DURATION} - A1 - A2 - A3$

Furthermore, the selection of the nasal, consonant or transitional to be changed is found by bounding consecutive segments of nasals and/or consonants by a stop, vowel, fricative, or burst at either end. One segment within these boundaries is selected for possible change in each case. If one of the boundary segments is a vowel, then for each of the segments, including the boundary segments, a sum $\frac{\text{DURATION} + A1 + A2 + A3}{8}$ is computed and saved.

If a nasal or consonant is a mild local maximum, then its sum will be greater than that of segments on either side of it and it will be the selected segment and be given a feasibility number. If the bounding segments are both non-vowels, then each segment within the boundary is given a weight =

$3 \cdot \text{DURATION} + A1 + A2$, and the segment with the greatest weight is selected and given a feasibility value.

If a nonstressed vowel is short or has a low amplitude defined by the conditions $A1 \leq 45$ or $5 \cdot \text{DURATION} + A1 \leq 75$, it is a candidate for becoming a consonant. Its feasibility value is:

Reference [1]: $90 - A1 - A2 - A3 - \text{DURATION}/20$

Computer program: $5 \cdot \text{DURATION} + A1 + A2 + A3 - 90$

If a fricative segment is short or does not have excellent unvoiced fricative characteristics (defined by the conditions $\text{DURATION} \leq 8$; or $5 \cdot \text{DURATION} + Z3 \leq 110$; or $Z3 \leq 70$ and $A1 + \frac{A2 \cdot 32}{A3} \geq 5$; or $Z3 \leq 70$ and either the segment is not the last or next-to-last segment or the $\text{DURATION} > 12$), it is a candidate for becoming a burst. Its feasibility value is

Reference [1]: $\frac{90 - \frac{\text{DURATION}}{20} - Z3}{3} + \frac{A3-A1}{2}$

Computer program: $\frac{5 \cdot \text{DURATION} + Z3 - 90}{3} + \frac{A3-A1}{2}$

A burst segment with a duration greater than 40 ms is a candidate for becoming a fricative. Its feasibility value is:

Reference [1]: $\frac{\frac{\text{DURATION}}{20} + Z3 - 90}{3} + \frac{A1-A3}{2}$

Computer Program: $\frac{90 - 5 \cdot \text{DURATION} - Z3}{3} + \frac{A1-A3}{2}$

If we consider that the feasibility values in [1] might have durations given in milliseconds and those in the program are given by 1 unit = 10 milliseconds, the differences in the treatment of duration are explained partially but are still basically different.

3.5 SIMILARITY EVALUATION PROCEDURE

The similarity evaluation subroutine, EVAL, uses subroutine GRAD to do the actual closeness calculation. In fact, EVAL proceeds mainly by setting up calls to GRAD. Because the subroutines GRAD and EVAL differ from Vicens's ALGOL program listing [1], they are described in detail here.

3.5.1 Subroutine GRAD

GRAD is called primarily from EVAL but is also called at various times by subroutine MAP to compute a closeness calculation. The input parameters to GRAD, set by the calling program, are:

MTEMP	--corresponds to INFLIM given in [1]
MDIV	--not in ALGOL listing in [1]--appears to have been added later
XMUL	--corresponds to weight given in [1]
RMAX	--corresponds to RATIO LIM given in [1]
II1	--the segment 1 parameter value
II2	--the segment 2 parameter value

The purpose of GRAD is to compute the closeness between II1 and II2 given MTEMP, MDIV, XMUL, and RMAX. The FORTRAN subroutine GRAD is given in Figure 2.

```
SUBROUTINE GRAD  
  
IMPLICIT INTEGER (A-Z)  
  
REAL XMUL, RMAX, RATIO  
  
IF (III.LT.MTEMP) III = III + MTEMP  
IF (II2.LT.MTEMP) II2 = II2 + MTEMP  
  
DIFF = IABS (III - II2)  
  
GSUM = 100  
  
IF (MAX0((III+II2)/MDIV,MTEMP)*GE*DIFF) RETURN  
  
GSUM = -200  
  
RATIO = (FLOAT(DIFF)*XMUL)/SQRT(FLOAT(III+II2))  
  
IF (RATIO*GT*RMAX) RETURN  
  
GSUM = IFIX(1.0-RATIO)*110.0  
  
IF (GSUM*GT*100) GSUM = 100  
IF (GSUM.LT.-200) GSUM = -200  
  
RETURN  
  
END
```

Figure 2. GRAD FORTRAN Subroutine

3.5.2 Subroutine EVAL

EVAL computer a similarity evaluation score between a segment in the R1 matrix and a segment in the R2 matrix. This score is adaptive with respect to the type of segments to be matched. It is composed of the following sections, each of which evaluates the type combinations appearing opposite

the section name. The type combination consists of a pair such as stop-stop, stop-burst, etc., where the first entry is the R2 type and the second entry is the R1 type.

<u>Section</u>	<u>Type Combinations</u>
EV100	stop-stop, stop-burst, burst-stop
EV200	consonant-stop, nasal-stop, stop-consonant, stop-nasal
EV300	consonant-vowel, nasal-vowel, vowel-consonant, vowel-nasal, vowel-vowel
EV400	stop-fricative, burst-burst, fricative-burst
EV500	consonant-burst, nasal-burst, burst-consonant, burst-nasal, burst-vowel, vowel-burst
EV600	consonant-consonant
EV700	consonant-fricative, nasal-fricative, stop-vowel, fricative-consonant, fricative-nasal, fricative-vowel, vowel-stop, vowel-fricative
EV800	burst-fricative, fricative-burst, fricative-fricative
EV900	nasal-nasal
EV1000	consonant-nasal
EV1100	nasal-consonant

EVAL is called with two input parameters, SEGR1 and SEGR2. SEGR1 is a segment number in the R1 matrix and SEGR2 is a segment number in the R2 matrix.

EVAL begins by setting:

J1 = SEGR1

J2 = SEGR2

I1 = TYPE1(J1)

I2 = TYPE2(J2)

If I1 is a vowel, set I1 = 6
If I2 is a vowel set, I2 = 6
If I1 is a transitional, set I1 = 1 (consonant)
If I2 is a transitional, set I2 = 1 (consonant)
and computing ETEMP = (I2*6) + I1 - 6 to find proper place to evaluate J1,
J2 on basis of types I1, I2.

EV100: STOP-STOP, STOP-BURST, BURST-STOP

Step 1 Set: II1 = DUR1(J1)
II2 = DUR2(J2)
RMAX = 1.5
XMUL = .625
MDIV = 10
MTEMP = 2
CALL GRAD
SCORE = GSUM

Step 2 Set: II1 = A11(J1)
II2 = A12(J2)
RMAX = 4.0
XMUL = .578125
CALL GRAD
SCORE = SCORE + GSUM

Step 3 Set: II1 = A21(J1)
II2 = A22(J2)
MTEMP = 4
CALL GRAD
XTEMP = 3
If A31(J1) + A32(J2) ≤ 6

4 October 1971

30

System Development Corporation
TM-4652/500/00

then: $SCORE = \frac{SCORE + GSUM}{XTEMP}$

and RETURN

else: $SCORE = SCORE + GSUM$

Step 4 Set: $II1 = Z31(J1)$
 $II2 = Z32(J2)$
 CALL GRAD
 $XTEMP = XTEMP + 1$
 $SCORE = \frac{SCORE + GSUM}{XTEMP}$
 then RETURN

EV200: CONSONANT-STOP, NASAL-STOP, STOP-CONSONANT, STOP-NASAL

Set: $II1 = A11(J1)$
 $II2 = A12(J2)$
 $MTEMP = 2$
 $MDIV = 10$
 $RMAX = 2.0$
 $XMUL = .625$
 CALL GRAD
 $SCORE = GSUM$
 If $SCORE < 0$ then return
 else go to EV100

EV300: CONSONANT-VOWEL, NASAL-VOWEL, VOWEL-CONSONANT, VOWEL-NASAL,
 VOWEL-VOWEL

$RMAX1 = RMAX2 = 0$
 $XMUL1 = XMUL2 = .1$
 $DIV1 = 4$
 $DIV2 = 2$

4 October 1971

31

System Development Corporation
TM-4652/500/00

EV350

If $\frac{|A11(J1) + A21(J1) - A12(J2) - A22(J2)|}{2} \leq 12$

then: $MTEMP = \frac{A11(J1) + A21(J1) + A12(J2) + A22(J2)}{4}$

else: $MTEMP = \frac{\text{MAX0}(A11(J1) + A21(J1), A12(J2) + A22(J2)) - 10}{2}$

$RMAX1 = RMAX1 + \frac{280.0 - \text{FLOAT}(2 \cdot MTEMP)}{100.0}$

$RMAX2 = RMAX2 + \frac{300.0 - \text{FLOAT}(MTEMP)}{80.0}$

$DIV1 = DIV1 + \frac{80 + MTEMP}{13}$

$DIV2 = DIV2 + \frac{100 + MTEMP}{13}$

$XMUL1 = XMUL1 + \frac{\text{FLOAT}(MTEMP) + 40.0}{140.0}$

$XMUL2 = XMUL2 + \frac{\text{FLOAT}(MTEMP \cdot 2) + 40.0}{240.0}$

Step 1 Set: $I11 = DUR1(J1)$

$I12 = DUR2(J2)$

$RMAX = 1.5$

$XMUL = .625$

$MDIV = 10$

$MTEMP = 2$

Call GRAD

$SCORE = GSUM$

4 October 1971

32

System Development Corporation
TM-4652/500/00

Step 2 Set: $II1 = Z11(J1)$
 $II2 = Z12(J2)$
 $RMAX = RMAX1$
 $XMUL = XMUL1$
 $MDIV = MDIV1$
 $MTEMP = 1$
 Call GRAD
 $SCORE = SCORE + (3 \cdot GSUM)$

Step 3 Set: $II1 = A11(J1)$
 $II2 = A12(J2)$
 $RMAX = RMAX2$
 $XMUL = XMUL2$
 $MTEMP = 2$
 CALL GRAD
 $SCORE = SCORE + \frac{GSUM}{2}$

Step 4 Set: $II1 = \frac{A21(J1) \cdot 32}{A11(J1)}$
 $II2 = \frac{A22(J2) \cdot 32}{A12(J2)}$
 Call GRAD
 $SCORE = SCORE + GSUM$

4 October 1971

33

System Development Corporation
TM-4652/500/00

Step 5 Set: $II1 = \frac{A31(J1) \cdot 32}{A11(J1)}$
 $II2 = \frac{A32(J2) \cdot 32}{A12(J2)}$

MTEMP = 4

Call GRAD

 $SCORE = SCORE + \frac{JSUM}{2}$
XTEMP = 6

Step 6 If $A21(J1) < 8$ and $A22(J2) < 8$ go to Step 7

 $III1 = Z21(J1)$
 $II2 = Z22(J2)$

RMAX = RMAX1 + .1

XMUL = XMUL1 - .1

MDIV = DIV1-2

MTEMP = 2

Call GRAD

 $SCORE = SCORE + 2 \cdot GSUM$

XTEMP = XTEMP+2

4 October 1971

34

System Development Corporation
TM-4652/500/00

Step 7 If A31(J1) < 8 and A32(J2) < 8 go to Step 8

III1 = Z31(J1)

III2 = Z32(J2)

RMAX = RMAX1 + .2

XMUL = XMUL1 - .2

MDIV = DIV1 - 3

MTEMP = 4

all GRAD

SCORE = SCORE + GSUM

XTEMP = XTEMP+1

Step 8 $\text{Score} = \frac{\text{SCORE}}{\text{XTEMP}}$

then return

4 October 1971

35

System Development Corporation
TM-4652/500/00

EV400: STOP-FRICATIVE, BURST-BURST, FRICATIVE-BURST

Step 1 Set: $II1 = DUR1(J1)$

$II2 = DUR2(J2)$

$RMAX = 1.5$

$XMUL = .625$

$MDIV = 10$

$MTEMP = 2$

Call GRAD

(Note: In the original computer program, the results of Step 1 of EV400 are not saved.)

Step 2 Set: $II1 = Z11(J1)$

$II2 = Z12(J2)$

Call GRAD

$SCORE = \text{Max}(25, GSUM)$

Step 3 Set: $II1 = A11(J1)$

$II2 = A12(J2)$

Call GRAD

$SCORE = SCORE + GSUM$

Step 4 Set: $II1 = \frac{Z31(J1)}{2}$

$II2 = \frac{Z32(J2)}{2}$

$MTEMP = 4$

Call GRAD

$SCORE = SCORE + GSUM$

4 October 1971

36

System Development Corporation
TM-4652/500/00

Step 5 Set: II1 = A31(J1)
II2 = A32(J2)
MTEMP = 2
Call GRAD
SCORE = Max (40, $\frac{\text{SCORE} + \text{GSUM}}{5}$)
Then return

EV500: CONSONANT-BURST, NASAL-BURST, BURST-CONSONANT, BURST-NASAL, BURST-VOWEL,
VOWEL-BURST

Step 1 Set: II1 = A11(J1)
II2 = A12(J2)
RMAX = 1.5
XMUL = .650
MDIV = 10
MTEMP = 2
Call GRAD
SCORE = GSUM
If SCORE < 0 then return
else: If Min (A11(J1), A12(J2)) < 4 go to EV400
else go to EV300

EV600: CONSONANT-CONSONANT

RMAX1 = .3
RMAX2 = .5
XMUL1 = 0
XMUL2 = 0
DIV1 = 0
DIV2 = 0
Go to EV350

4 October 1971

37

System Development Corporation
TM-4652/500/00

EV700: CONSONANT-FRICATIVE, NASAL-FRICATIVE, STOP-VOWEL, FRICATIVE-CONSONANT,
FRICATIVE-NASAL, FRICATIVE-VOWEL, VOWEL-STOP, VOWEL-FRICATIVE

SCORE = - 1000

RETURN

EV800: BURST-FRICATIVE, FRICATIVE-BURST, FRICATIVE-FRICATIVE

Step 1 Set: $II1 = DUR1(J1)$

$II2 = DUR2(J2)$

$RMAX = 1.5$

$XMUL = .625$

$MDIV = 10$

$MTEMP = 2$

Call GRAD

$SCORE = GSUM$

$DIV1 = 20$

If $SCORE \geq 50$ then $DIV1 = 50$

Step 2 Set: $II1 = \frac{Z31(J1)}{2}$

$II2 = \frac{Z32(J2)}{2}$

$RMAX = 4.0$

$MTEMP = 4$

Call GRAD

$SCORE = SCORE + 2 \cdot GSUM$

4 October 1971

38

System Development Corporation
TM-4652/500/00

Step 3 Set: $II1 = A11(J1)$
 $II2 = A12(J2)$
 $MTEMP = 2$
Call GRAD
 $SCORE = SCORE + GSUM$

Step 4 Set: $II1 = \frac{A31(J1)}{2}$
 $II2 = \frac{A32(J2)}{2}$
Call GRAD
 $SCORE = MAX \left(DIV1, \frac{SCORE + GSUM}{5} \right)$
RETURN

EV900: NASAL-NASAL

If $DUR1(J1) > 5$ and $DUR2(J2) > 5$ go to EV600

else:

Step 1 Set: $II1 = A11(J1)$
 $II2 = A12(J2)$
 $RMAX = 2.0$
 $XMUL = .375$
 $MDIV = 10$
 $MTEMP = 2$
Call GRAD
 $SCORE = GSUM$
If $SCORE < 0$, return.
else go to EV600

4 October 1971

39

System Development Corporation
TM-4652/500/00

EV1000: CONSONANT-NASAL

If DUR1(J) < 5 or Z11(J1) > 5 or A21(J1)·12 ≥ A11(J1)
or A31(J1)·10 ≥ A11(J1)
then go to EV600
else go to EV1110

EV1100: NASAL-CONSONANT

If DUR2(J2) < 5 or Z12(J2) > 5 or A22(J2)·12 ≥ A12(J2)
or A32(J2)·10 ≥ A12(J2)
then go to EV600
else go to EV1110

EV1110

Step 1 Set: I11 = A11(J1)
II2 = A12(J2)
RMAX = 2.0
XMUL = .35
MDIV = 10
MTEMP = 2
Call GRAD
SCORE = GSUM
If SCORE < 0 then return.
Else go to EV600

4 October 1971

40
(last page)

System Development Corporation
TM-4652/500/00

REFERENCES

- [1] Vicens, P., Aspects of Speech Recognition by Computer, Stanford University AI Memo. No. 85 (CS1277), 1969.
- [2] Kameny, I. and H. B. Ritea, Description and Analysis of the Vicens-Reddy Preprocessing and Segmentation Algorithms, System Development Corporation, Technical Memorandum TM-4652/200/00, 4 December 1970, 63 pp.
- [3] Kameny, I. and H. B. Ritea, Description and Analysis of the Vicens-Reddy Recognition Algorithms, System Development Corporation, Technical Memorandum TM-4652/300/00, 29 March 1971, 29 pp.
- [4] Kameny, I. The Lexicon Design for the IBM 360/67, System Development Corporation, Technical Memorandum TM-4652/400/00, 28 May 1971, 15 pp.